How Good is Recursive Bisection?

Horst D. Simon

NASA Ames Research Center
Moffett Field, CA 94035

Shang-Hua Teng*

Department of Mathematics
Massachusetts Institute of Technology
Cambridge, MA 02139

August 18, 1993

Abstract

The most commonly used p-way partitioning method is recursive bisection. It first "optimally" divides the graph (mesh) into two equal sized pieces and then recursively divides the two pieces. We show that, due to the greedy nature of and the lack of global information, recursive bisection, in the worst case, may produce a partition that is very far away from the optimal one. Our negative result is complemented by two positive ones: First we show that for some important classes of graphs that occur in practical applications, such as well shaped finite element and finite difference meshes, recursive bisection is normally within a constant factor of the optimal one. Secondly, we show that if the balanced condition is relaxed so that each block in the partition is bounded by $(1+\epsilon)n/p$, then there exists an approximately balanced recursive partitioning scheme that finds a partition whose cost is within an $O(\log p)$ factor of the cost of the optimal p-way partition.

Keywords: Communication cost, data and computation mapping on parallel machines, load balancing, mesh partitioning, parallel processing, recursive bisection, scalable parallel algorithms, well-shaped finite element and difference meshes.

1 Introduction

Graph partitioning is one of the most important problems for solving large-scale scientific computing problems on a massively parallel machine. A key subroutine in such parallel

*This work was performed when the author was at NASA Ames Research Center, Moffett Field, CA 94035.

How Good is Recursive Bisection?

Horst D. Simon[1] and Shang-Hua Teng[2]
Report RNR-93-012, August 1993

NAS Systems Division
Applied Research Branch
NASA Ames Research Center, Mail Stop T045-1
Moffett Field, CA 94035

(submitted to *SIAM Journal on Scientific Computing)*

[1]The author is an employee of Computer Sciences Corporation. This work was supported through NASA Contract NAS 2-12961.

[2]Dept. of Mathematics, Massachusetts Inst. of Technology, Cambridge, MA 02139. This work was performed when the author was visiting NASA Ames Research Center, and was supported through NASA Contract NAS 2-12961 with Computer Sciences Corporation

processing is to map irregular and unstructured computations onto a distributed memory parallel machine to achieve load balance and to reduce communication cost[2, 27, 6, 28, 32]. In particular, to process a computational task, which is represented by a graph G whose edges give the communication pattern and whose nodes represent the computations, on a parallel machine of p processors, we need to decompose G into p subgraphs $G_1, ..., G_p$ and assign the computation of each subgraph to a processor. To achieve "optimal" speed-up, ideally, we need a p-way partition (see next section for a formal definition) such that $|G_i| = n/p$ and the communication cost is minimized, where the communication cost is modeled by the total number (or weights) of edges whose endpoints are in different components. Thus, the partitioning problem has applications in both direct and iterative methods for sparse linear and non-linear systems [5, 6, 13, 14, 12, 15, 20, 21, 22, 24, 27]. It is also an important subproblem in run-time and compiler-time optimization for high performance parallel processing [4, 6] and for VLSI layout [18].

The most commonly used approach for p-way partitioning is to recursively bisect the graph[1], i.e., first optimally divide the graph (mesh) into two equal sized pieces and then recursively divide the two pieces. Some extended heuristics have been proposed that apply quadsectioning or octsectioning in place of bisecting [16]. Preliminary experimental results seems to indicate that quadsectioning and octsectioning, though more expensive than bisecting, find better p-way partitions. Little is known, however, about how good indeed is recursive bisection and whether more global optimization schemes should be sought.

In this paper, we show that, due to the greedy nature of and the lack of global information, recursive bisection may, in the worst case, produce a partition that is very far away from being optimal. In other words, optimal recursive bisection may not lead to a good p-way partition. Our results hold even for sparse graphs and more structured graphs such as planar graphs and geometric graphs [22]. The negative result also extends to quadsectioning and octsectioning.

On the optimistic side, our negative result is complemented by two positive results.

First, we show that for some important classes of graphs that occur in practical applications, such as well shaped finite element and finite difference meshes [3, 7, 22, 23, 26], recursive bisection is within a constant factor of the optimal one in the expected case. In particular, it follows from a result of [22] that recursive bisection finds a p-way partition of cost $O(p^{1/d}n^{1-1/d})$ for well shaped meshes embedded in d dimensions.

Secondly, we show that if we relax the balance condition to be that each block in the partition is bounded by $(1+\epsilon)n/p$, then there exists an approximately balanced recursive partitioning algorithm (see Section 6) that finds a partition whose cost is within an $O(\log p)$ factor of the cost of the optimal p-way partition. Our result implies that it may not be a good idea to insist upon the perfect bisection condition at each step of the recursive bisection scheme.

[1]When p is not a power of 2, some simple variant of recursive bisection is used.

Section 2 introduces the definitions and notations that will be used. Section 3 gives a dense graph and a sparse graph for whom optimal 4-way partition has cost 12 and recursive bisection has cost $\Omega(n^2)$ and $\Omega(n)$, respectively. Section 4 gives a tight bound of $\Theta(n^2/p^2)$ and $\Theta(n/p)$, respectively for dense graphs and sparse graphs, on the approximation ratio of recursive bisection. Section 5 shows that for well-shaped meshes in d dimensions, recursive bisection always finds a p-way partition of cost $O(p^{1/d}n^{1-1/d})$. Section 6 introduces the notion of approximately-balanced p-way partition and gives a recursive partitioning algorithm that is within an $O(\log p)$ factor of the cost of the optimal p-way partition.

2 Definitions

A *bisection* of a graph G is a division of its vertex set into two subsets A and B of exactly equal sizes (we assume that the graph has even number of vertices). The *cost* of a bisection is the number of edges one of whose endpoint is in A and another is in B. A *p-way partition* of a graph G is a division of its vertex set into p subsets each of size n/p. The *cost* of a p-way partition is the number of edges whose endpoints are in different subsets.

When p is a power of 2, a p-way partition can be found by recursively applying bisection.

Algorithm (*Recursive Bisection*)
Input: (a graph G of n vertices and an integer p, Assume $K = n/p$).
Output: (a p-way partition of G).

1. Find an "optimal" bisection G' and G'' of G;
2. If $|G'| > K$ then
   - Recursive Bisection(G');
   - Recursive Bisection(G'');
3. Return the subgraphs $G_1, ..., G_p$ so obtained;

Note that the problem of finding an optimal bisection itself is *NP*-hard [9]. Recursive Bisection as given above is a template of practical implementations where Step 1 is replaced by the best available bisection algorithm. Our results can be extended to the case when Step 1 is implemented as an approximately optimal bisection algorithm.

For a $1/2 \leq \delta \leq 1$, a *δ-bisection* (or a *δ-edge-separator*) is a partition of a graph G into two subgraphs G_1 and G_2 such that both $|G_1| \leq \delta|G|$ and $|G_2| \leq \delta|G|$. The *cost* of a δ-bisection is the number of edges between G_1 and G_2. So, a bisection can be viewed as a more restricted set of edge-separators where the splitting ratio is exactly $1/2$.

We distinguish two classes of graphs: dense graphs and sparse graphs. A dense graph may have $O(n^2)$-edges and while a sparse graph has only $O(n)$ edges. We can further restrict that each vertex in a sparse graph has a bounded degree. As shown in [30], most well shaped finite-element meshes in three dimensions are sparse.

A p-way partition algorithm has *approximation ratio* $\alpha : \alpha \geq 1$, if for each graph G, it finds a p-way partition of cost at most α times the cost of an optimal p-way partition. Associated with recursive bisection is a tree, called the *partition tree*. Notice that the height of the partition tree is $\log p$.

3 4-Way Partition

We first consider a 4-way partition and show that there exist graphs that admit a constant costed 4-way partition, but on which the recursive bisection produces a partition of cost $\Omega(n^2)$ in the dense case and $\Omega(n)$ in the sparse case.

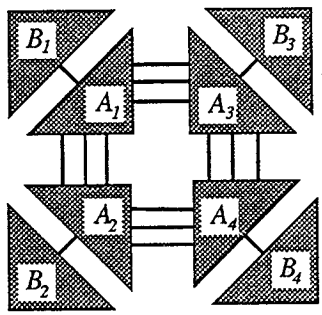

Figure 1: An example of 4-way partition where the optimal cost is 12 and recursive bisection has cost $\Theta(n^2)$ in the dense case and $\Theta(n)$ in the sparse case.

The outlook of our graphs is shown in Figure 1. It is a graph of n nodes (assuming n is a power of 2), where A_i has $(1/8 + \epsilon_i)n$ nodes and B_i has $(1/8 - \epsilon_i)n$ nodes, where ϵ_i's $(1 \leq i \leq 4)$ satisfy the following conditions.

1. $-1/8 < \epsilon_i < 1/8$ and $\epsilon_i \neq 0$;
2. $\epsilon_1 + \epsilon_2 + \epsilon_3 + \epsilon_4 = 0$; and
3. There are no $i, j \in \{1, 2, 3, 4\}$ such that $\epsilon_i + \epsilon_j = 0$.

It is not hard to see that such ϵ_i's exist (e.g., choose ϵ_1, ϵ_2, ϵ_3 at random and choose ϵ_4 so that condition 2 holds. Then with high probability, condition 3 holds as well).

In the dense case, we simply let A_i and B_i be cliques; while in the sparse case, we choose A_i and B_i to be sparse expanders. All balanced edge-separators of a clique of m nodes have

cost $\Omega(m^2)$ and all balanced edge-separators of a sparse expander of m nodes have cost $\Omega(n)$. One way to construct an expander is to choose a random bounded degree graph, and it follows from a result of Erdös, Graham, Szemerědi that all balanced edge-separators of almost all such linear sized graphs have cost $\Omega(n)$ [8].

The optimal 4-way partition is the decomposition of the graph into $A_i \cup B_i$, $1 \leq i \leq 4$. The total cut size is clearly 12. In contrast, the recursive bisection decomposes the graph into $A_1 \cup A_2 \cup A_3 \cup A_4$ and $B_1 \cup B_2 \cup B_3 \cup B_4$. But then, because of condition 3, at least one block of $A_1 \cup A_2 \cup A_3 \cup A_4$ ($B_1 \cup B_2 \cup B_3 \cup B_4$) will be divided two pieces whose sizes are of constant fraction of the original block in the next level of recursive bisection. Because A_i's and B_i's are cliques in the dense case and expanders in the sparse case, this dividing phase will introduce a cost of $\Omega(n^2)$ in the dense case and $\Omega(n)$ in the sparse case to the final partition. Hence the cost of the 4-way partition of recursive bisection is at least $\Omega(n^2)$ in the dense case and $\Omega(n)$ in the sparse case.

4 p-Way Partitions

The 4-way partition example can be used to construct a tight lower bound on the approximation ratio of recursive bisection. Let us first consider the example graph in Figure 2. It is an illustration of a $p = 64$-way partition.

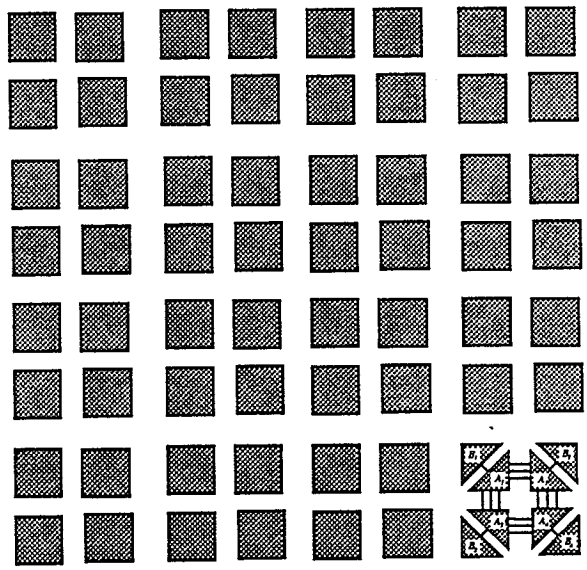

Figure 2: An example of p-way partition where optimal cost is 12 and recursive bisection cost is $\Theta(n^2/p^2)$ in the dense case and $\Theta(n/p)$ in the sparse case.

The graph contains 60 disconnected blocks of equal size (n/p) and a graph from the last section of size $4n/p$. Clearly, the optimal partition has cost 12, which is the cost of decomposing the subgraph from the last section.

Once again, we have both dense and sparse cases. In the first $\log p - 2$ levels, recursive bisection simply decomposes the graph into blocks of four each. The cost so far is zero. Therefore, the subgraph from the last section stays as one of the blocks. Then, it becomes clear that recursive bisection yields a partition of cost of $\Theta(n^2/p^2)$ in the dense case and $\Theta(n/p)$ in the sparse case.

Our next example is given in Figure 3, which is graph of 16 copies of the graph from the last section.

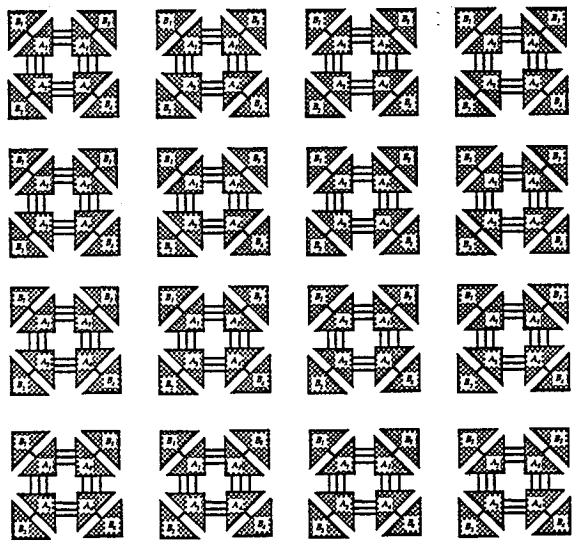

Figure 3: An example of p-way partition where the optimal cost is p and recursive bisection cost is $\Theta(n^2/p)$ in the dense case and $\Theta(n)$ in the sparse case.

Notice that in this case, the optimal partition (into $p = 64$ pieces) has cost $3p$, while the partition of recursive bisection has cost $\Theta(n^2/p)$ in the dense case and $\Theta(n)$ in the sparse case.

Interestingly, in both examples, the approximation ratio of the recursive bisection is $\Theta(n^2/p^2)$ in the dense case and $\Theta(n/p)$ in the sparse case. Does recursive bisection always achieve approximation ratio no worse than $O(n^2/p^2)$ and $O(n/p)$, respectively? We now show that the answer is "yes".

Lemma 4.1 *Recursive bisection has worst case approximation ratio of* $\Theta(n^2/p^2)$ *in the dense case and* $\Theta(n/p)$ *in the sparse case for p-way partitions.*

Proof: We just give the proof for the sparse case. The proof for the dense case is similar. The cost of the p-way partition of recursive bisection is bounded from above by the total number of edges. Hence, by our assumption, it is $O(n)$.

Thus, if the cost of the optimal p-way partition is $\Omega(p)$, then recursive bisection clearly has approximation ratio $O(n/p)$.

Now, suppose the cost of the optimal p-way partition is $k < p$. Let $C_1, C_2, ..., C_p$ be the blocks in such an optimal partition. So the number of blocks that is connected to some other blocks, called *connecting blocks*, is at most k. Recursive bisection has zero cost except when the subgraph containing those connecting blocks. So, it generates a p-way partition of cost at most $O(kn/p)$. □

5 Graphs with a Family of Small Edge Bisectors

Many graphs in practical applications have small separators [22], i.e., they are graphs that have the property that every subgraph of then of m vertices has an $f(m)$-bisector, for a sub-linear function f. We call such a property the *family of $f(m)$-bisection property.* How good is recursive bisection?

Figure 4 illustrates the partition tree of a p-way partition given by recursive bisection and the upper bound on the cost of each node.

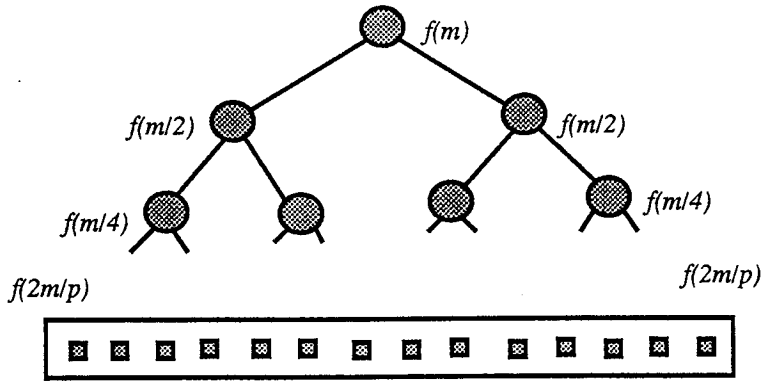

Figure 4: Partition with small edge bisectors

The next lemma follows Immediately from Figure 4.

Lemma 5.1 *If G has the property that every subgraph of G of m vertices has an $f(m)$-bisector, then the p-way partition generated by recursive bisection has cost*

$$\sum_{i=0}^{\log p-1} 2^i f(n/2^i)$$

The condition of family of $f(m)$-bisection property can be relaxed to the following condition of *family of $f(m)$-separator property*: every subgraph of G of m vertices has an $f(m)$-edge separator whose removal divide the graph into two disconnected subgraphs G_1 and G_2, such that $\max(|G_1|, |G_2|) \leq \delta n$, for some constant δ. We say such an edge separator δ-splitts G. Using an argument due to Lipton and Tarjan [19] and Gilbert [12], we can show

that if a graph has a family of $f(m)$-separators, then it has a family of $O(f(m))$-bisectors. Here is a list of graphs that have a family of small separators (and hence bisectors).

- **Planar graphs** [19] have a family of $O(\sqrt{m})$-separators.
- **Bounded genus graphs** [10] have a family of $O(\sqrt{gm})$-separators, where g is the genus.
- **Bounded minor graphs** [1] have a family of $O(h^{1.5}\sqrt{m})$-separators, where h is the size of the largest minor clique.
- **Well shaped meshes** [22] have a family of $O(m^{1-1/d})$-separators, where d is the dimension of the space in which the mesh is embedded.
- **k-nearest neighbor graphs** [22] have a family of $O(k^{1/d}m^{1-1/d})$-separators.

The following lemma is a simple consequence of Lemma 5.1,

Lemma 5.2 *If $f(m) = m^{1-1/d}$, then the p-way partition generated by recursive bisection has cost $O(p^{1/d}n^{1-1/d})$.*

Proof: The cost of the p-way partition is bounded from above by

$$\begin{aligned} & \sum_{i=0}^{\log p-1} 2^i (n/2^i)^{1-1/d} \\ = \; & n^{1-1/d}\Big(\sum_{i=1}^{\log p-1} 2^{i/d}\Big) \\ = \; & O(p^{1/d}n^{1-1/d}) \end{aligned}$$

□

It follows from [30] that most well-shaped meshes (in d-dimensions) in applications have no p-way partition of size $o(p^{1/d}n^{1-1/d})$. So the p-way partition of recursive bisection is optimal (up to a constant factor).

We can extend the results of Sections 3 and 4 to graphs discussed in this section. In particular,

Theorem 5.3 *Recursive bisection has worst case approximation ratio of $\Theta(\sqrt{n/p})$ for planar graphs and $\Theta((n/p)^{1-1/d})$ for well-shaped meshes in d dimensions.*

6 Approximately Balanced p-Way Partition

Note that even though one can use general edge separators (not necessarily bisection, [11, 19, 22]) or use *minimum quotient separators* of [17] at each level of a recursive partitioning algorithm, there is no guarantee on the approximation ratio as long as the final partition is required to be a (perfectly balanced) p-way partition. All of the results in the previous sections generalize.

The question now becomes: can we trade the balanced condition for a better approximation ratio of the recursive partitioning scheme?

Let $\beta \geq 1$ be a real number. A (β, p)-*way partition* of a graph G is a decomposition of G into $G_1, ..., G_p$ such that $|G_i| \leq \beta|G|/p$, for all $1 \leq i \leq p$. Thus, p-way partition is a $(1, p)$-way partition. The *cost* of a (β, p)-way partition is the number of edges of G whose two endpoints are not in the same subgraph.

Recall that a δ-bisection, $1/2 \leq \delta < 1$, is a partition of a graph G into two subgraphs G_1 and G_2 such that both $|G_1| \leq \delta|G|$ and $|G_2| \leq \delta|G|$. The cost of a δ-bisection is the number of edges between G_1 and G_2.

We first show that the following standard recursive partitioning scheme finds a $(2, p)$-way partition with cost at most $O(\log p)$ times the cost of the optimal p-way partition.

Algorithm (*Recursive Partitioning*)
Input: (a graph G of n vertices and an integer p).
Output: (a $(2, p)$-way partition of G).

1. Let $K = n/p$;
2. Let $G_1, ..., G_h$ be the h subgraphs obtained from running the subroutine Recursive Cutting(G,K) below;
3. If $h \leq p$, then return $(G_1, ..., G_h)$ else repeatedly merge the smallest two subgraphs until p subgraphs remain.

Subroutine (*Recursive Cutting*(G, K))

1. Let $s = |G|/K$;
2. Find an optimal $(1/2 + 1/s)$-bisection G' and G'' of G;
3. If $|G'| > 2K$ then Recursive Cutting(G',K);
4. If $|G''| > 2K$ then Recursive Cutting(G'', K);

We will use the following simple lemma to prove the Theorem 6.2 below.

Lemma 6.1 *Suppose $X = \{x_1, ..., x_m\}$ is a set of positive real numbers such that $0 \leq x_i \leq 1$. Then X can be divided into two subsets X_1 and X_2 such that $|\sum_{x \in X_1} x - \sum_{y \in X_2} y| \leq 1$.*

Proof: We use the greedy approach: First, we put all elements from X in a queue and maintain two sets that are initially empty; secondly, assign the largest element from the queue to the set with smaller sum, and delete the element from the queue, until the queue is empty. Clearly, the sums of the two sets so constructed differ at most by one. □

Theorem 6.2 *Let G be a graph and p be a positive integer. If the cost of the optimal p-way partition is C, then Recursive Partitioning finds a $(2, p)$-way partition of cost $O(C \log p)$.*

Proof: The basic idea of the proof is to argue that the cost induced at each level of the partitioning tree is at most C. Because the partitioning tree has $O(\log p)$ levels, the theorem then follows.

Clearly, the partition associated with the root of the partitioning tree has cost at most C. This can be shown by the following argument, more complex than needed, but we will use it to bound the cost of other levels.

Let $B_1, ..., B_p$ be the subgraphs in an optimal p-way partition of G. After removing all the inter-block edges, we can group $B_1, ..., B_p$ into two subsets of equal size. This shows that G has a $(1/2 + 1/p)$-bisection, in fact a perfect bisection, of size at most C.

Note that $|B_i| = K = n/p$. Now, we look at the ith level of the partitioning tree, and we show that there exists an approximately balanced bisection for each node at level i so that the total cost is at most C. Because Recursive Partitioning uses optimal approximately balanced bisection for each node, the total cost at level i can only be smaller, and hence is bounded by C.

Implicitly, $B_1, ..., B_p$ may be decomposed into pieces in the previous $i - 1$ levels. So, each node of the partitioning tree has a subgraph that have some subsets of these pieces of B_i's. Note that the size of those pieces is at most K. Once again, image that we delete all the active inter-block edges at level i. The cost of these edges in total is at most C. Now, each node at level i contains some subset of these pieces of B_i's, each of which has size at most K.

The partitioning problem of each node can be viewed as a bin-packing or a two processor job scheduling problem, i.e., we can apply Lemma 6.1 to divide the pieces of each node into two groups. By Lemma 6.1, if the subgraph at a node at level i has size sK, then the larger group has size at most $sK/2 + K$. Hence the grouping gives a $(1/2 + 1/s)$-bisection.

Note that after deleting the inter-block edges, we did not remove any other edges, and hence the cost at level i is at most C. □

In conjunctions with the result in [17], we have the following corollary.

Corollary 6.3 *There is a polynomial time algorithm that finds a $(2,p)$-way partition of cost $O(C \log n \log p)$, where C is the cost of the optimal p-way partition.*

If we choose the $K = \epsilon n/p$ in Step 1 of Recursive Partitioning, then we have the following strengthened result.

Theorem 6.4 *Let G be a graph and p be a positive integer. If the cost of the optimal p-way partition is C, then Recursive Partitioning finds a $(1+\epsilon,p)$-way partition of cost $O(C \log p)$. Also, there exists a polynomial time algorithm that finds a $(1+\epsilon,p)$-way partition of cost $O(C \log n \log p)$.*

7 Final Remarks

Our results of Sections 3 and 4 can be extended to the case when recursive quadsectioning or octsectioning is used. However, it gives some theoretical evidence to the experimental claim that recursive quadsectioning and octsectioning usually find a better p-way partition. The results in Section 5 are mainly observational and follow quite directly from the previous separator results [1, 10, 19, 22]. These results give an absolute upper bound on the cut-size of p-way partition. They show that the ratio of cut-size to the graph size is $O((p/n)^{1/d}) < 1$. So the ratio of computations to communication in processing well-shaped meshes is reasonably balanced as p and especially n increase. This demonstrates that the partitioning based parallel algorithms are scalable. The results of Section 6 gives a theoretical justification to the recursive approach taken in [27, 6, 11, 22, 30] and many similar heuristics currently implemented. We expect to see these ideas extended for better, perhaps more global, schemes, for approximating p-way partitioning.

Acknowledgement: We would like to thank David Bailey and Jim Ruppert for their careful proof-reading of the draft and helpful comments.

References

[1] N. Alon, P. Seymour, and R. Thomas. A separator theorem for graphs with an excluded minor and its applications. In *ACM STOC*, 293–299. 1990.

[2] M. J. Berger and S. Bokhari. A partitioning strategy for nonuniform problems on multiprocessors. *IEEE Trans. Comp.*, C-36:570-580, 1987.

[3] M. Bern, D. Eppstein and J. R. Gilbert. Provably good mesh generation. In *31st Annual Symposium on Foundations of Computer Science, IEEE*, 231–241, 1990, (to appear JCSS).

[4] H. Berryman, J. Saltz and J. Scroggs. Execution time support for adaptive scientific algorithms on distributed memory machines. *Currency: Practice and Experience* 3(3), pp159–178, June 1991.

[5] P. E. Bjørstad and O. B. Widlund. Iterative methods for the solution of elliptic problems on regions partitioned into substructures. *SIAM J. Numer. Anal.*, 23:1097-1120, 1986.

[6] G. E. Blelloch, A. Feldmann, O. Ghattas, J. R. Gilbert, G. L. Miller, D. R. O'Hallaron, E. J. Schwabe, J. R. Shewchuk and S.-H. Teng. Automated parallel solution of unstructured PDE problems. *CACM*, to appear, 1993.

[7] L. P. Chew. Guaranteed quality triangular meshes, Department of Computer Science, Cornell University TR 89-893, 1989.

[8] P. Erdós, R. L. Graham, and E. Szemerédi. On sparse graphs with dense long paths. *Comp. and Math. with Appl.* 1: 365–369, 1975.

[9] M. R. Garey and D. S. Johnson. *Computer and Intractability: a guide to the theory of NP-completeness.* Freeman, San Francisco, 1979.

[10] J. R. Gilbert, J. P. Hutchinson, and R. E. Tarjan. A separation theorem for graphs of bounded genus. J. Algorithms, 5, 391–407, 1984.

[11] J. R. Gilbert, G. L. Miller, and S.-H. Teng. Geometric mesh partitioning: implementation and experiments. Technical Report, Xerox Palo Alto Research Center, to appear.

[12] J. R. Gilbert and R. E. Tarjan. The analysis of a nested dissection algorithm. *Numerische Mathematik*, 50(4):377–404, 1987.

[13] J. A. George. Nested dissection of a regular finite element mesh. *SIAM J. Numerical Analysis*, 10: 345–363, 1973.

[14] J. A. George and J. W. H. Liu. *Computer Solution of Large Sparse Positive Definite Systems.* Prentice-Hall, 1981.

[15] S. Hammond and R. Schreiber. Solving unstructured grid problems on massively parallel computers. Technical Report TR 90.22, RIACS, 1990.

[16] B. Hendrickson and R. Leland. An improved spectral graph partitioning algorithm for mapping parallel computations. Technical Report, Sandia Lab, 1992.

[17] F. T. Leighton and S. Rao. An approximate max-flow min-cut theorem for uniform multicommodity flow problems with applications to approximation algorithms. In *29th Annual Symposium on Foundations of Computer Science*, pp 422-431, 1988.

[18] C. E. Leiserson. *Area Efficient VLSI Computation.* Foundations of Computing. MIT Press, Cambridge, MA, 1983.

[19] R. J. Lipton and R. E. Tarjan. A separator theorem for planar graphs. *SIAM J. of Appl. Math.*, 36:177–189, April 1979.

[20] R. J. Lipton, D. J. Rose, and R. E. Tarjan. Generalized nested dissection. *SIAM J. on Numerical Analysis*, 16:346–358, 1979.

[21] J. W. H. Liu. The solution of mesh equations on a parallel computer. in 2nd Langley Conference on Scientific Computing, 1974.

[22] G.L. Miller, S.-H. Teng, W. Thurston, and S.A. Vavasis. Automatic Mesh Partitioning. To appear in *Proceedings of the 1992 Workshop on Sparse Matrix Computations: Graph Theory Issues and Algorithms*, Institute for Mathematics and its Applications.

[23] S. A. Mitchell and S. A. Vavasis. Quality mesh generation in three dimensions. Proc. ACM Symposium on Computational Geometry, pp 212-221, 1992.

[24] V. Pan and J. Reif. Efficient parallel solution of linear systems. In *Proceedings of the 17th Annual ACM Symposium on Theory of Computing*, pages 143–152, Providence, RI, May 1985. ACM.

[25] A. Pothen, H. D. Simon, K.-P. Liou. Partitioning sparse matrices with eigenvectors of graphs. *SIAM J. Matrix Anal. Appl.* 11 (3), pp 430–452, July, 1990.

[26] J. Ruppert. *Results on Triangulation and High Quality Mesh Generation.* Ph.D. Thesis, University of California at Berkeley, 1992.

[27] E. Schwabe, G. Blelloch, A. Feldmann, O. Ghattas, J. Gilbert, G. Miller, D. O'Hallaron, J. Schewchuk and S.-H. Teng. A separator-based framework for automated partitioning and mapping of parallel algorithms in scientific computing. In *First Annual Dartmouth Summer Institute on Issues and Obstacles in the Practical Implementation of Parallel Algorithms and the use of Parallel Machines*, 1992.

[28] H. D. Simon. Partitioning of unstructured problems for parallel processing. *Computing Systems in Engineering* 2:(2/3), pp135-148.

[29] G. Strang and G. J. Fix. *An Analysis of the Finite Element Method*, Prentice-Hall, 1973.

[30] S.-H. Teng. *Points, Spheres, and Separators: A Unified Geometric Approach to Graph Partitioning.* PhD thesis, School of Computer Science, Carnegie Mellon University, Pittsburgh, 1991. CMU-CS-91-184.

[31] P. M. Vaidya. Constructing provably good cheap preconditioners for certain symmetric positive definite matrices. *IMA Workshop on Sparse Matrix Computation: Graph Theory Issues and Algorithms*, Minneapolis, Minnesota, October 1991.

[32] R. D. Williams. Performance of dynamic load balancing algorithms for unstructured mesh calculations. Technical Report, California Institute of Technology, 1990.